I-SHEEP: SELF-ALIGNMENT OF LLMS THROUGH AN ITERATIVE SELF-ENHANCEMENT PARADIGM

Yiming Liang[1,2,6*] **Xingwei Qu**[4,5*] **Tianyu Zheng**[5*] **Jiawei Guo**[5] **Xinrun Du**[5] **Zhenzhu Yang**[5]
Jiaheng Liu[5] **Chenghua Lin**[4] **Ge Zhang**[3,5†] **Lei Ma**[6,7†] **Wenhao Huang**[7†] **Jiajun Zhang**[1,2†]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]Institute of Automation, Chinese Academy of Sciences, [3]University of Waterloo
[4]The University of Manchester, [5]M-A-P, [6]BAAI, [7]Peking University
`gezhang@umich.edu, lei.ma@pku.eud.cn, rubio8741@gmail.com, jjzhang@nlpr.ia.ac.cn`

ABSTRACT

Large Language Models (LLMs) have achieved significant advancements, however, the common learning paradigm treats LLMs as passive information repositories, neglecting their potential for active learning and alignment. Some approaches train LLMs using their own generated synthetic data, exploring the possibility of active alignment. However, there is still a huge gap between these one-time alignment methods and the continuous automatic alignment of humans. In this paper, we introduce **I-SHEEP**, an **I**terative **S**elf-En**H**anc**E**m**E**nt **P**aradigm. This human-like paradigm enables LLMs to **continuously self-align from scratch with nothing**. Compared to the one-time alignment method Dromedary (Sun et al., 2023b), which refers to the first iteration in this paper, I-SHEEP can significantly enhance capacities on both Qwen and Llama models. I-SHEEP achieves a maximum relative improvement of 78.2% in the Alpaca Eval, 24.0% in the MT Bench, and an absolute increase of 8.88% in the IFEval accuracy over subsequent iterations in Qwen-1.5 72B model. Additionally, I-SHEEP surpasses the base model in various standard benchmark generation tasks, achieving an average improvement of 24.77% in code generation tasks, 12.04% in TrivialQA, and 20.29% in SQuAD. We also provide new insights based on the experiment results. Our codes, datasets, and models are available at https://anonymous.4open.science/r/SHEEP/.

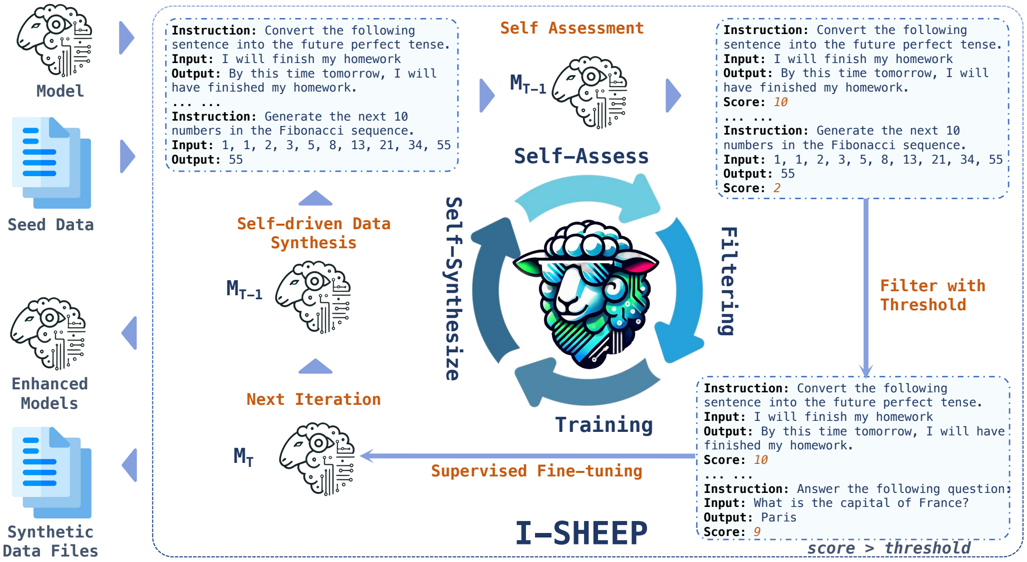
1 INTRODUCTION

Large Language Models (LLMs) have achieved significant success, yet they remain far from achieving the autonomous self-regulation and coherent self-understanding characteristic of human-like Artificial General Intelligence (AGI). In the pretraining phase, LLMs learn and memorize common knowledge from vast amounts of raw text. During the Supervised Fine-Tuning (SFT) phase, LLMs are trained with question-answer pairs to develop instruction-following capabilities. Both phases currently treat LLMs as passive information repositories, neglecting their potential for active learning and alignment. Furthermore, aligning a base model from scratch requires substantial high-quality data across various tasks and scenarios. This data collection, annotation, and cleaning process is labor-intensive, time-consuming, and must ensure diversity and representativeness. Humans can automatically answer relevant questions after acquiring knowledge (Khader et al., 2016), and there is no need for an explicit process like SFT to teach humans how to answer questions. Therefore, exploring active, automatic, and continuous alignment from scratch, akin to human learning, is a crucial step for LLMs toward achieving AGI.

There are several approaches developed to align base models in low-resource scenarios, aiming to reduce or eliminate reliance on human supervision signals (Wang et al., 2022b; Sun et al., 2023b;a; Xu et al., 2024b). For instance, Self Instruct and Dromedary (Wang et al., 2022b; Sun et al., 2023b) actively improve themselves using self-generated data, while Magpie (Xu et al., 2024b) aligns the base model by prompting more powerful models to generate instruction-output pair data. Although these methods demonstrate some level of proactivity, there remains a significant gap between these

---

* Equal contribution † Corresponding author



Instruction: Convert the following sentence into the future perfect tense.
Input: I will finish my homework
Output: By this time tomorrow, I will have finished my homework.
... ...
Instruction: Generate the next 10 numbers in the Fibonacci sequence.
Input: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
Output: 55

Instruction: Convert the following sentence into the future perfect tense.
Input: I will finish my homework
Output: By this time tomorrow, I will have finished my homework.
Score: *10*
... ...
Instruction: Generate the next 10 numbers in the Fibonacci sequence.
Input: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
Output: 55
Score: *2*

Instruction: Convert the following sentence into the future perfect tense.
Input: I will finish my homework
Output: By this time tomorrow, I will have finished my homework.
Score: *10*
... ...
Instruction: Answer the following question:
Input: What is the capital of France?
Output: Paris
Score: *9*

Figure 1: Pipeline of I-SHEEP. The I-SHEEP framework takes the base model and small seed dataset as input, aligns the base model iteratively from scratch independently, and finally obtains the self-enhanced models and high-quality synthetic datasets. The I-SHEEP framework consists of four main components: the self-synthesize process generates instruction-pair data, the self-assessment assesses the quality of the resulting data, the filtering component filters out low-quality data based on self-assessment, and the training component integrates the high-quality data into the base model.

one-time alignment processes and the continuous, automatic alignment for various environments seen in human learning.

Educational research suggests that metacognitive self-assessment plays a vital role in continuous alignment, helping students reflect on their knowledge and skills, manage cognitive resources, and improve their performance (Yan et al., 2023). Inspired by this perspective, we explore a new human-like paradigm for LLMs, I-SHEEP, designed to enable LLMs to proactively, automatically, and continuously align from scratch, even with minimal external signal. As shown in Figure **??**, I-SHEEP begins with seed data and leverages LLMs powerful understanding and generation capabilities to create additional instruction-output pair data. We then perform self-assessment, allowing the model to monitor and assess its learning process. By filtering out incorrect cognitions and retaining accurate ones, LLMs can self-align by training themselves with these correct cognitions. Through iterative repetition of this process, the model can continuously and autonomously align from scratch, relying solely on its internal knowledge.

The main contributions can be summarized as follows: **(1) We propose I-SHEEP, a human-like learning paradigm for LLMs, enabling active, automatic, and continuous self-alignment from scratch using only their internal knowledge.** Based on the one-time alignment approach, I-SHEEP explores the incorporation of metacognitive self-assessment to monitor and manage the learning process as well as continuous self-alignment from scratch. **(2) We integrate metacognitive self-assessment into the alignment process, allowing the model to monitor and manage its learning process.** Experiments explore how varying levels of metacognitive abilities impact the efficiency of the I-SHEEP framework. **(3) We find that the self-enhancement potential is closely related to the metacognitive self-assessment level and model size.** We conducted experiments to explore the performance of I-SHEEP on different model sizes and varying levels of metacognitive self-assessment.

2 RELATED WORK

2.1 AUTOMATIC DATA SELECTION

Zhou et al.; Bai et al. emphasize that dataset quality outweighs quantity during the instruction fine-tuning stage. As a result, some studies on instruction data selection have emerged, focusing on identifying high-quality subsets from candidate datasets(Li et al., 2023a; Du et al., 2023; Liu et al., 2023; Li et al., 2024; Ge et al., 2024; Xia et al., 2024). These methods aim to improve the model performance, accelerate the training process, and facilitate data-efficient alignment. Li et al. introduce an Instruction-Following Difficulty (IFD) metric and use it to select the top 5% of data for fine-tuning models. The filtering phase in the I-SHEEP framework does not rely on carefully selected metrics, external models or human assistance. Additionally, our method is orthogonal to these selection approaches.

2.2 SYNTHETIC DATA FOR IMPROVING MODEL

Generating synthetic data refers to using the powerful generative capabilities of LLMs to create new data that simulates potential real-world scenarios, reducing the need for costly manual labeling. Some methods use the model's self-generated data to improve itself (Wang et al., 2022b; Sun et al., 2023b;a; Yehudai et al., 2024). Other methods leverage powerful closed models to generate synthetic data, enhancing the capabilities of open-source models (Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023a; Yu et al., 2023; Wei et al., 2023). In addition to generating complete instruction-output pairs, some methods collect existing raw data and synthesize corresponding questions or answers to create supervised data for improving the model (Huang et al., 2022; Li et al., 2023b; Zheng et al., 2024b; Mitra et al., 2024; Wang et al., 2022a; Asai et al., 2023). Some methods begin with instruction-output pairs, generating feedback or refining answers to improve data quality and enhance the model's reasoning capabilities.(Lu et al., 2023; Li & He, 2024; Gou et al., 2023). The I-SHEEP framework evolves from the aforementioned static, one-time improvement paradigm to a dynamic, continuous self-enhancement process.

2.3 ITERATIVE ENHANCEMENT FOR LLMS

There are several approaches to iterative enhancement that rely on the help of strong models or external tools (Chen et al., 2024; 2023; Lu et al., 2023; Gao et al., 2023; Lee et al., 2024). IterAlign (Chen et al., 2024) employs strong models like GPT-4 and Claude2 to detect and correct errors in responses from base LLMs and give the corresponding constitution for improving the safety of LLMs. These methods in iterative enhancement typically depend on strong models or external tools to guarantee ongoing model optimization and avoid model collapse. In addition, some methods explore iterative enhancement in the RLHF phase to continuously align the model with human preference (Yuan et al., 2024; Liu et al., 2024; Pang et al., 2024; Xu et al., 2024a; 2023b; Wu et al., 2024; Wang et al., 2024). These iterative RLHF methods start with the aligned model, while we focus on the base model continuous self-alignment from scratch.

3 METHODOLOGY

3.1 SELF-DRIVEN DATA SYNTHESIS

Self Instruct (Wang et al., 2022b) leverages an off-the-shelf large language model (LLM) for the generation of synthetic data. The approach starts with a small set of 175 prompts, known as the seed task pool, leveraging the model's powerful understanding and generative capabilities to generate more prompts and responses. This section elaborates on the Self-Driven Data Synthesis process from two perspectives: Instruction generation and response generation. For ease and consistency in data creation, we utilize a standardized instruction format introduced by Alpaca (Taori et al., 2023), enabling the direct generation of instructions along with their corresponding potential inputs.

Instruction generation. Having some prompts from the seed dataset D^s and the meta-prompt p^{meta} from Alpaca (Taori et al., 2023). The process that model M generating new prompt set $\mathcal{P}$ through In-Context Learning (ICL) can be modeled as:

$$p_i = argmax_p(p_i|\{d\}, p^{meta}; \theta)$$

p_i denotes a new prompt generated by model M, $\{d\}$ represents a subset sampled from the seed dataset D^s for in-context learning (ICL). The symbol θ stands for the parameter of model M.

Response generation. After obtaining the set of prompts $\mathcal{P}$, we use the model M to generate corresponding responses $\mathcal{R}$ via a zero-shot approach.

3.2 SELF-ASSESSMENT AND DATA FILTERING

To ensure that the data used for self-enhancement maintains a high-quality standard, a two-stage process comprising self-assessment and data filtering is implemented.

Self-Assessment. We pair the generated prompt set $\mathcal{P}$ and response set $\mathcal{R}$ to form the instruction-output pair data D_{raw}. Given the capacity limitations of models, ensuring the quality of synthetic pairs can be challenging, making it essential to assess the quality of the generated data. Manual assessment is often impractical, therefore, we introduce an automated assessment method that relies solely on the model. Specifically, the model autonomously evaluates each generated response for its quality and adherence to the instructions. Each entry is scored based on predefined criteria, which quantitatively reflect the compliance and quality of the response.

Data Filtering. After the self-assessment, the subsequent data filtering phase discards entries that do not meet the specified quality threshold. This step guarantees that only entries of the highest quality are retained in the dataset, thereby enhancing the overall reliability and utility of the generated data. Initially, we apply heuristic rule-based filtering to the generated data during data generation, following the Self-Instruct (Wang et al., 2022b). Additionally, after data generation, we filter the instruction-output pairs based on the assessment scores from the self-assessment phrase. A threshold $\mathcal{C}$ is applied to filter D_{raw} based on assessment scores, yielding a high-quality dataset D.

3.3 ITERATIVE CONTINUOUS MODEL ENHANCEMENTS

The Iterative Self-Enhancement algorithm aims to incrementally enhance a language model by generating and utilizing high-quality synthetic datasets. As shown in Algorithm 1, starting with an initial model M^{base} and a small seed task set D^s, the algorithm iterates over a specified number of steps $\mathcal{T}$ and a filtering threshold $\mathcal{C}$. At each iteration t, the algorithm performs several functions: it generates a new set of prompts, $\mathcal{P}^t$, using a prompt generation process that leverages the current model M^t and the seed data D^s. It then produces corresponding responses, $\mathcal{R}^t$, forming a raw dataset, $D^t_{raw} = \{\mathcal{P}^t, \mathcal{R}^t\}$. This dataset undergoes a self-assessment process to evaluate the quality of responses, after which it is filtered using the threshold $\mathcal{C}$ to retain only high-quality data, resulting in D^t. The model M^t is then trained on D^t to align it closely with the refined data, enhancing its performance iteratively by supervised fine-tuning (SFT) approach. This process continues until it concludes at step $\mathcal{T}$, ultimately producing a stronger language model $M^{\mathcal{T}}$ and a refined synthetic dataset $D^{\mathcal{T}}$.

Algorithm 1: Iterative Self-Enhancement Algorithm

Input: Initial seed task set D^s, Base model M^{base}

Hyper-parameter: Iteration steps $\mathcal{T}$, Filtering threshold $\mathcal{C}$, Data size $\mathcal{I}$

Output: Enhanced LLMs $M^{\mathcal{T}}$, High-quality datasets $D^{\mathcal{T}}$

```
Initialize $M^0 \leftarrow M^{base}$
for $t = 0$ to $\mathcal{T}$ do
    $\mathcal{P}^t \leftarrow$ generate_prompts($D^s$, $p^{meta}$, $M^t$)
    $\mathcal{R}^t \leftarrow$ generate_responses($\mathcal{P}^t$, $M^t$)
    $D^t_{raw} \leftarrow \{(\mathcal{P}^t, \mathcal{R}^t)\}$
    $S^t \leftarrow$ self_assessment($D^t_{raw}$, $M^t$)
    $D^t \leftarrow$ filtering($D^t_{raw}$, $S^t$, $\mathcal{C}$)
    $M^{t+1} \leftarrow$ SFT($M^{base}$, $D^t$)
end for
return $M^t, D^t$
```

4 EXPERIMENTS

4.1 EVALUATION

Chat Evaluation. We evaluate the instruction-following ability and response quality of aligned models with three chat benchmarks, AlpacaEval(Dubois et al., 2023), MT-Bench(Zheng et al., 2024a), and IFEval(Zhou et al., 2023), due to their comprehensiveness, fine granularity, and reproducibility. Both AlpacaEval and MT-Bench rely on GPT as an evaluator. IFEval provides four types of accuracy scores: prompt-level strict-accuracy, inst-level strict-accuracy, prompt-level loose-accuracy, and inst-level loose-accuracy.

OpenCompass Evaluation. We use the OpenCompass evaluation platform (Contributors, 2023), a comprehensive one-stop platform for LLM evaluation. The evaluation includes standard benchmarks such as BoolQ Clark et al. (2019), PIQA (Bisk et al., 2019), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-c (Clark et al., 2018), OpenBookQA-Fact (Mihaylov et al., 2018), CommonsenseQA (Contributors, 2023), and MMLU (Hendrycks et al., 2020). It also includes code generation benchmarks such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), word knowledge benchmark TriviaQA (Joshi et al., 2017), and reading comprehension benchmark SQuAD2.0 (Rajpurkar et al., 2018). Full results on these benchmarks are available in Appendix D.

4.2 MAIN SETTINGS

We conduct experiments on the Qwen-1.5 (Team, 2024) and Llama-3 (Dubey et al., 2024) models to validate the effectiveness and generalization of I-SHEEP. Additionally, we explore the impact of different model sizes on I-SHEEP by conducting experiments on Qwen-1.5 1.8B, 4B, 7B, 14B, 32B, and 72B models, providing a detailed analysis based on the experimental results. In each iteration, the dataset for training is generated by the model from the last iteration. The case study of the generated data and the overall quality analysis can be found in Appendix C and Appendix G, respectively. We utilized LLaMA-Factory (Zheng et al., 2024c) for LoRA fine-tuning, with specific parameters detailed in Appendix F. Under the configuration of using VLLM for inference (Kwon et al., 2023), the maximum duration of each iteration is about 4 hours on NVIDIA A800-SXM4-80GB$\times$8, equivalent to one iteration time for Qwen-1.5 72B.

4.3 SELF-ASSESSMENT AND FILTER SETTINGS

During the self-assessment phase, we propose three variants, simple standard prompt, combined standard prompt, and ICL prompt, to evaluate data quality. Detailed prompt contents can be found in Appendix A.

In the filtering phase, there are six settings, simple standard prompt based filtering, combined standard prompt based filtering, ICL prompt filtering, PerPLexity (PPL) filtering, density filtering, and the combination of density and PPL filtering. In addition to the first three filtering settings based on scores obtained in the Self-Assessment phase, we also explore data filtering methods that do not rely on external tools or models. For example, PPL filtering uses the PPL value computed by the model itself to evaluate the quality of instruction-output pairs, thereby eliminating low-quality data. We filter out data points with PPL greater than 50. Density filtering extracts vector representations from the model's final layer and performs K Nearest Neighbors (KNN) clustering, sampling from each cluster to ensure dataset diversity. We set 3000 as the clustering number K. The combination of density and PPL filtering setting first clusters the data and then selects samples with lower PPL values from each cluster, ensuring the filtered dataset's quality and diversity.

4.4 BASELINE

We use the base model, Self Instruct (Wang et al., 2022b), and Dromedary (Sun et al., 2023b) as baselines to explore the continuous and automatic enhancement of the human-like framework, I-SHEEP. Self Instruct is a one-time alignment approach where LLMs are trained directly on data they generate, without a self-assessment phase. Similarly, Dromedary is a one-time alignment process where the model generates responses following specific principles, which are then engraved into the model. This approach is similar to the first iteration setting described in this paper.

4.5 ITERATIVE SETTINGS AND ABLATION SETTINGS

Iterative Settings. We investigate the impact of I-SHEEP on efficiency across different iterative self-enhancement settings, including using data generated by the last iteration model to train the base model, using data generated by the last iteration model to train the last iteration model, and using data generated by all previous iterations to train the base model. Additionally, we directly generate 20K and 30K data points for comparative experiments to eliminate the influence of data size in the iterative settings mentioned above. Notably, in the first iteration, all settings are identical, where the base model generates 10k data, filters it, and uses it to fine-tune itself, akin to the Dromedary(Sun et al., 2023b).

Ablation Settings. We adjust high-dimensional variables such as the threshold $\mathcal{C}$ in the self-assessment phase, data size $\mathcal{I}$ in the generation phase, and iteration steps $\mathcal{T}$ in the iterative training phase to validate their impact on I-SHEEP. Furthermore, we conduct ablation experiments with different levels of metacognitive self-assessment, including no self-assessment, assessing only response quality, assessing only instruction-following degree, and assessing both response quality and instruction-following degree.

5 RESULTS

5.1 MAIN RESULTS

Table 1: Main results: experimental performance of various model sizes across different iteration steps. We stop the iteration when the performance improvement in subsequent iterations stagnates or diminishes. The red settings represent the baseline for our experiments on Qwen-1.5 72B. The Self Instruct (Wang et al., 2022b) setting involves training the model using generated data without filtering. The iter1 setting indicates training the model using filtered data, which is selected based on prompts, similar to the Dromedary approach (Sun et al., 2023b). **Bold** results indicate the best performance for each model size at different iteration steps. P-S, I-S, P-L, and I-L denote prompt-level strict accuracy, instance-level strict accuracy, prompt-level loose accuracy, and instance-level loose accuracy, respectively.

Setting		**Chat Benchmark**						**Standard Benchmark**			
		Alpaca Eval	MT Bench	IFEval				Code		Knowledge	Reading Comprehension
				P-S	I-S	P-L	I-L	Human Eval/Plus	MBPP	Trivia QA	SQuAD 2.0
1.8B	base	–	–	–	–	–	–	6.71/6.10	16.40	**31.18**	**30.02**
	iter1	1.51	**3.76**	15.53	25.30	17.74	28.06	11.59/9.15	16.80	19.38	13.16
	iter2	1.54	3.53	**16.27**	**27.10**	**19.22**	**31.41**	**15.24/12.20**	17.40	16.88	14.57
	iter3	**2.30**	3.16	13.68	24.46	15.34	27.22	14.02/10.98	**17.80**	12.49	13.91
4B	base	–	–	–	–	–	–	10.98/8.54	28.00	**40.95**	27.96
	iter1	2.61	4.97	19.41	29.98	24.03	34.77	30.49/26.83	34.00	38.94	24.90
	iter2	2.96	4.79	**19.78**	**32.61**	**23.84**	**36.81**	31.10/27.44	35.20	37.20	24.63
	iter3	**3.78**	**4.99**	18.85	31.41	22.18	35.37	**32.93/28.66**	**35.80**	35.37	**31.67**
7B	base	–	–	–	–	–	–	10.98/8.54	36.60	**51.00**	33.14
	iter1	5.19	5.08	28.47	39.93	31.05	43.41	45.73/39.63	**41.20**	45.81	26.36
	iter2	**5.37**	**5.13**	**30.13**	**40.89**	**33.09**	**43.88**	**47.56/42.68**	41.00	42.83	28.36
	iter3	5.22	4.97	29.21	40.29	30.68	43.05	45.12/40.24	40.60	40.53	**33.76**
14B	base	–	–	–	–	–	–	17.68/15.85	41.40	**57.72**	20.37
	iter1	4.77	5.68	28.84	41.13	**33.46**	**46.40**	45.73/40.85	**49.00**	56.81	30.52
	iter2	6.27	**5.97**	**30.87**	42.93	**33.46**	**46.40**	**48.78/42.07**	45.60	54.45	38.57
	iter3	**7.30**	5.48	30.13	**43.05**	33.27	46.04	**50.00/43.29**	45.20	55.30	**43.42**
32B	base	–	–	–	–	–	–	22.56/21.34	**47.40**	65.88	29.56
	iter1	8.27	5.56	33.46	45.32	37.52	50.12	**58.54/51.83**	44.20	**60.81**	41.34
	iter2	8.26	5.68	36.04	47.60	**39.56**	**51.92**	56.71/50.61	41.80	59.43	42.15
	iter3	**9.30**	**5.69**	**36.41**	**47.96**	38.82	51.56	56.71/51.83	42.20	59.73	44.04
	iter4	8.64	5.62	33.83	46.88	38.45	51.56	56.10/50.61	40.60	58.95	**47.07**
72B	iter1	6.64	6.43	35.67	49.16	40.48	53.96	50.61/45.12	51.20	60.81	50.68
	iter2	9.06	7.90	37.34	51.32	40.85	54.56	56.71/49.39	51.80	61.55	52.27
	iter3	10.51	**7.97**	41.22	54.32	44.18	57.19	56.10/50.61	52.60	62.00	61.42
	iter4	11.22	5.45	42.14	54.56	46.21	58.63	51.83/47.56	**56.00**	**70.43**	64.55
	iter5	**11.83**	5.62	**44.55**	**55.88**	**47.50**	**58.75**	**56.71/53.66**	55.60	70.11	**67.95**
	iter6	11.60	5.75	42.33	53.84	45.10	56.95	51.22/48.17	55.20	70.01	67.82
Base Model		–	–	–	–	–	–	21.34/20.12	50.20	58.07	47.66
Self Instruct		5.26	7.82	33.64	47.60	39.56	53.00	53.05/46.95	48.40	71.25	51.90

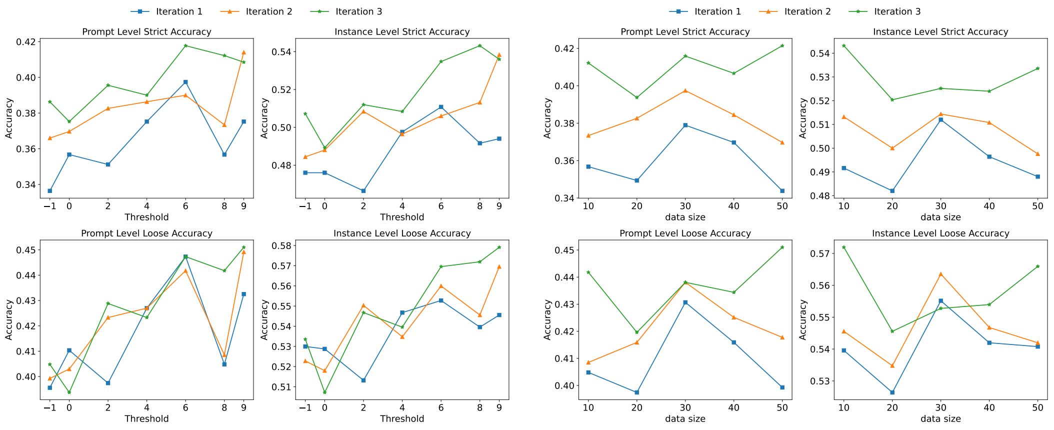
Table 1 shows the experimental performance of various model sizes across different iteration steps. There are some new findings: **(1) I-SHEEP exhibits efficacy across various model sizes, with particularly notable improvements in 72B.** I-SHEEP achieves a maximum relative improvement of 78.2% in the Alpaca Eval, 24.0% in the MT Bench, and an absolute increase of 8.88% in the IFEval prompt-level strict accuracy over subsequent iterations in Qwen-1.5 72B model. Additionally, I-SHEEP surpasses the base model in various standard benchmark generation tasks, achieving an average improvement of 24.77% in code generation tasks, 12.04% in Trivial QA, and 20.29% in SQuAD. we find that the scores for the second round of dialogues drop significantly after the fourth iteration. This decline is likely due to our generated data consisting solely of single-round dialogues, which do not improve and may even harm the scores for the second round of dialogues. More analysis can be found in the Appendix E. **(2) The potential for improvement varies with different model sizes.** The 1.8B, 4B, 7B, and 14B models exhibit improvements over two iterations, 32B and 72B model can improve three and five iterations, respectively, according to the IFEval benchmark.

5.2 ITERATIVE SETTING RESULTS

Table 2 presents the chat benchmark performance for the Qwen-1.5 72B model across various iteration settings. More benchmark results are available in Appendix D. Our findings are as follows: **(1) Training the base model with data from the last iteration model is effective for iterative self-enhancement.** At the third iteration in the One_base Setting, training the base model with the last iteration data achieves the highest performance on the chat benchmark. The notable performance improvement under this setting suggests that the model has the potential for further enhancement (refer to Table 1 72B results). Therefore, we chose the *One_base* setting for all subsequent experiments. **(2) The data size is not the main factor influencing iterative improvement.** Training the base model with the last iteration data at the 3rd iteration outperforms training the base model with a combination of all data from previous iterations.

Table 2: The performance of various iteration settings at different iteration steps. *One_base* and *One_last* means using data from the last iteration to train the base and the last iteration model respectively. *Total_base* means using data from all previous iterations to train the base model. *Direct* represents using data generated by the base model to train itself.

Setting		**Chat Benchmark**					
		Alpaca Eval	MT Bench	IFEval			
				P-S	I-S	P-L	I-L
iter1(Dromedary)		6.64	6.43	35.67	49.16	40.48	53.96
Direct	20k	7.18	7.87	39.37	50.72	43.25	54.56
	30k	6.53	7.75	38.08	50.24	43.07	54.92
Total_base	iter2	7.25	7.94	39.00	50.72	45.47	56.47
	iter3	7.51	7.94	37.52	48.32	41.59	52.76
One_last	iter2	7.76	7.76	38.45	50.48	41.96	54.92
	iter3	8.45	7.82	38.63	51.80	42.70	56.12
One_base	iter2	9.06	7.90	37.34	51.32	40.85	54.56
	iter3	**10.51**	**7.97**	**41.22**	**54.32**	**44.18**	**57.19**



(a) Performance across various thresholds

(b) Performance across various data sizes

Figure 2: Ablation performance for the first three iterations across various thresholds and data sizes. In subfigure 2a, the threshold -1 means that the generated data is not filtered by heuristic rules. The threshold 0 represents that the I-SHEEP process does not use the self-assessment phase. Other thresholds represent filtering low-quality data using the threshold, which refers to the score from the self-assessment phase. In subfigure 2b, the values on the horizontal axis represent the amount of data generated (in thousands).

5.3 ABLATION ANALYSIS

As shown in Figure 2a, as the threshold increases, the performance of I-SHEEP at the 3rd iteration shows an upward trend. The threshold 8 is selected to ensure the possibility of further iterative improvement, given the significant performance increase in iteration 2 and iteration 3, and the good performance at iteration 3 with a threshold of 8. Choosing a threshold of 8 is not necessarily the optimal experimental setting, as thresholds of 6, 7, 8, and 9 are all possible.

Figure 2b shows a stable improvement in the first three iterations across different data sizes (10k, 20k, 30k, 40k, 50k), demonstrating the robustness of the I-SHEEP framework with respect to data size. When the data size is 10k, the model performs well in the 3rd iteration, meanwhile, there are significant improvements between the first iterations. Considering the above factors and resource savings, we chose 10k as the final data size setting.

5.4 METACOGNITIVE SELF-ASSESSMENT ANALYSIS

5.4.1 SELF-ASSESSMENT ROBUSTNESS ANALYSIS

Table 3 shows the performance of various self-assessment degrees in the first three iterations. See the Appendix D for more benchmark results. The following findings can be drawn from the table: **(1) Using explicit self-assessment prompt is better than using simple model internal states.** On all four IFEval accuracies, the highest values are obtained in the setting where the model is explicitly prompted for self-assessment. **(2) The I-SHEEP framework is robust to prompt.** Although the criteria differ between simple and combined standard prompt settings, their performance is quite similar. Even without designing a prompt, using just a few examples for ICL can achieve comparable results.

Table 3: Experimental results using different filtering methods that rely solely on the model. *PPL* filtering involves removing data points with high PPL values. *Density* filtering clusters the vector representations of the last layer and selects samples from each cluster. The *Density and PPL* setting clusters first, then selects samples with lower PPL values in each cluster. *Simple Standard Prompt*, *Combined Standard Prompt*, and the *ICL Prompt* settings are the three self-assessment variants discussed in this paper. Please refer to the appendix for detailed prompt content. **Bold results** indicate the best results, and blue results indicate the second-best results in each column.

Setting		**Chat Benchmark**			
		IFEval			
		P-level S-accuracy	I-level S-accuracy	P-level L-accuracy	I-level L-accuracy
Density	iter1	34.20	46.76	39.56	51.80
	iter2	37.34	49.76	41.22	53.72
	iter3	37.52	49.52	39.56	51.56
PPL	iter1	36.60	49.16	41.77	54.08
	iter2	36.04	46.64	39.92	50.84
	iter3	33.27	45.92	36.41	49.52
Density and PPL	iter1	37.52	49.64	42.51	54.68
	iter2	40.48	52.16	44.73	56.24
	iter3	38.82	50.48	41.96	53.60
Simple Standard Prompt	iter1	35.30	48.20	42.33	54.68
	iter2	36.23	49.28	40.67	53.60
	iter3	**42.14**	54.08	**45.10**	56.83
Combined Standard Prompt	iter1	35.67	49.16	40.48	53.96
	iter2	37.34	51.32	40.85	54.56
	iter3	41.22	**54.32**	44.18	**57.19**
ICL Prompt	iter1	38.82	49.40	43.99	55.04
	iter2	37.34	50.84	43.25	56.47
	iter3	41.22	53.72	43.99	36.12

5.4.2 SELF-ASSESSMENT LEVEL ANALYSIS.

As shown in Table 4, we explore the efficiency of I-SHEEP across various self-assessment levels. Our findings include the following key points: **(1) The higher the level of self-assessment, the greater the improvement in the efficiency and potential of the I-SHEEP framework.** Assessing both quality and instruction-following degree achieves the best performance at 3rd iteration, com-

Table 4: Experimental results across various self-assessment levels. The *no_prompt* setting means no metacognitive self-assessment. The *quality* setting assesses only the output quality. The *following* setting measures instruction adherence, and the *both* setting assesses both response quality and the degree of instruction adherence simultaneously. **Bold results** indicate the best results, and blue results indicate the second-best results in each column.

Setting	IFEval			
	P-level S-accuracy	I-level S-accuracy	P-level L-accuracy	I-level L-accuracy
no_prompt_iter1	35.67	47.60	41.04	52.88
no_prompt_iter2	36.97	48.80	40.30	51.80
no_prompt_iter3	37.52	48.92	39.37	50.72
quality_iter1	37.34	48.20	42.51	52.64
quality_iter2	36.04	49.04	40.67	53.00
quality_iter3	37.71	51.44	41.96	54.92
following_iter1	35.49	47.72	38.82	51.68
following_iter2	40.48	52.76	43.62	56.35
following_iter3	39.93	51.68	43.25	55.52
both_iter1	35.30	48.20	42.33	54.68
both_iter2	36.23	49.28	40.67	53.60
both_iter3	**41.14**	**54.08**	**45.10**	**56.83**

pared to the other settings. **(2) Evaluating the degree of instruction adherence of data pairs is better than only evaluating the quality of output.** Compared to the *quality* experimental group, the *following* experimental group achieved an overall victory at 2nd iteration on the IFEval benchmark.

5.5 GENERALIZATION OF I-SHEEP

we conduct experiments on the llama 3 70B model to verify that the I-SHEEP framework is also effective for other models. Table 5 shows that llama 3 is also stably and iteratively enhanced through the I-SHEEP framework. Moreover, the significant improvement between the 2nd iteration and the 3rd iteration indicates that llama3 has the potential for further enhancement.

Table 5: Performance in the first three iterations of llama3. ↑Green values are the improvements over the first iteration.

Setting	IFEval			
	P-level S-accuracy	I-level S-accuracy	P-level L-accuracy	I-level L-accuracy
llama3_iter1	9.43	19.06	10.35	21.70
llama3_iter2	9.61 ↑0.18	21.34 ↑2.28	11.28 ↑0.93	23.74 ↑2.04
llama3_iter3	12.38 ↑2.95	20.98 ↑1.92	14.42 ↑4.07	23.86 ↑2.16

6 CONCLUSION

In this paper, we emphasize and formally introduce a challenging task, continuous self-alignment with nothing, which aims to explore how to achieve and to what extent self-alignment can be realized. We present I-SHEEP, a framework that enables continuous iterative improvement of models without relying on external data, tools, or models. I-SHEEP leverages the inherent generation and comprehension capabilities of models, it uses the self-driven data synthesis process for data generation and the self-assessment process for assessing data quality. Based on these assessment scores, high-quality data is filtered and used to train the model itself. Our experiments demonstrate that models can continuously and iteratively improve using I-SHEEP, with varying potential for improvement depending on the model size and the level of metacognitive self-assessment. Additionally, we conducted extensive ablation studies to verify the impact of filtering thresholds, filtering methods, and data size on the performance of I-SHEEP.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv: 2310.11511*, 2023.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv: 2108.07732*, 2021.

Yuelin Bai, Xinrun Du, Yiming Liang, Yonggang Jin, Ziqiang Liu, Junting Zhou, Tianyu Zheng, Xincheng Zhang, Nuo Ma, Zekun Wang, Ruibin Yuan, Haihong Wu, Hongquan Lin, Wenhao Huang, Jiajun Zhang, Wenhu Chen, Chenghua Lin, Jie Fu, Min Yang, Shiwen Ni, and Ge Zhang. Coig-cqia: Quality is all you need for chinese instruction fine-tuning. *arXiv preprint arXiv: 2403.18058*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv: 1911.11641*, 2019.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv: 2107.03374*, 2021.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. Iterative translation refinement with large language models. *arXiv preprint arXiv: 2306.03856*, 2023.

Xiusi Chen, Hongzhi Wen, Sreyashi Nag, Chen Luo, Qingyu Yin, Ruirui Li, Zheng Li, and Wei Wang. Iteralign: Iterative constitutional alignment of large language models. *North American Chapter of the Association for Computational Linguistics*, 2024. doi: 10.48550/arXiv.2403.18341.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv: 1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv: 1803.05457*, 2018.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`, 2023.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. Mods: Model-oriented data selection for instruction tuning. *arXiv preprint arXiv: 2311.15653*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models. *arXiv preprint arXiv: 2407.21783*, 2024.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Neural Information Processing Systems*, 2023. doi: 10.48550/arXiv.2305.14387.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, and Jun Ma. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. *AAAI Conference on Artificial Intelligence*, 2023. doi: 10.48550/arXiv.2308.14034.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Hongxia Ma, Li Zhang, Hao Yang, and Tong Xiao. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. *arXiv preprint arXiv: 2402.18191*, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv: 2309.17452*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv: 2009.03300*, 2020.

Jiaxin Huang, S. Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *Conference on Empirical Methods in Natural Language Processing*, 2022. doi: 10.48550/arXiv.2210.11610.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv: 1705.03551*, 2017.

Patrick H Khader, Thorsten Pachur, Lilian AE Weber, and Kerstin Jost. Neural signatures of controlled and automatic retrieval processes in memory-based decision-making. *Journal of cognitive neuroscience*, 28(1):69–83, 2016.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipali, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv: 2403.15042*, 2024.

Long Li and Xuzheng He. How do humans write code? large models do it the same way too. *arXiv preprint arXiv: 2402.15729*, 2024.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv: 2308.12032*, 2023a.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv: 2402.00530*, 2024.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv: 2308.06259*, 2023b.

Jie Liu, Zhanhui Zhou, Jiaheng Liu, Xingyuan Bu, Chao Yang, Han-Sen Zhong, and Wanli Ouyang. Iterative length-regularized direct preference optimization: A case study on improving 7b language models to gpt-4 level. *arXiv preprint arXiv: 2406.11817*, 2024.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv: 2312.15685*, 2023.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. Self: Self-evolution with language feedback. *arXiv preprint arXiv: 2310.00533*, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, oct-nov 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL `https://aclanthology.org/D18-1260`.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv: 2402.14830*, 2024.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv: 2404.19733*, 2024.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv: 1806.03822*, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv: 1907.10641*, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv: 1904.09728*, 2019.

Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Salmon: Self-alignment with principle-following reward models. *arXiv preprint arXiv: 2310.05910*, 2023a.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *NEURIPS*, 2023b.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Qwen Team. Introducing qwen1.5, February 2024. URL `https://qwenlm.github.io/blog/qwen1.5/`.

Haoyu Wang, Guozheng Ma, Ziqiao Meng, Zeyu Qin, Li Shen, Zhong Zhang, Bingzhe Wu, Liu Liu, Yatao Bian, Tingyang Xu, Xueqian Wang, and Peilin Zhao. Step-on-feet tuning: Scaling self-alignment of llms via bootstrapping. *arXiv preprint arXiv: 2402.07610*, 2024.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, E. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*, 2022a. doi: 10.48550/arXiv.2203.11171.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *Annual Meeting of the Association for Computational Linguistics*, 2022b. doi: 10.48550/arXiv.2212.10560.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with oss-instruct. *arXiv preprint arXiv: 2312.02120*, 2023.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv: 2407.19594*, 2024.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv: 2402.04333*, 2024.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv: 2304.12244*, 2023a.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Iterative preference optimization with the pairwise cringe loss. *arXiv preprint arXiv: 2312.16682*, 2023b.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv: 2404.10719*, 2024a.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv: 2406.08464*, 2024b.

Z. Yan, E. Panadero, X. Wang, et al. A systematic review on students' perceptions of self-assessment: Usefulness and factors influencing implementation. *Educational Psychology Review*, 35:81, 2023. doi: 10.1007/s10648-023-09799-1. URL `https://doi.org/10.1007/s10648-023-09799-1`.

Asaf Yehudai, Boaz Carmeli, Yosi Mass, Ofir Arviv, Nathaniel Mills, Assaf Toledo, Eyal Shnarch, and Leshem Choshen. Genie: Achieving human parity in content-grounded datasets generation. *arXiv preprint arXiv: 2401.14367*, 2024.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, Yishujie Zhao, Wenxiang Hu, and Qiufeng Yin. Wavecoder: Widespread and versatile enhancement for code large language models by instruction tuning. *arXiv preprint arXiv: 2312.14187*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv: 2401.10020*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/P19-1472.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024a.

Tianyu Zheng, Shuyue Guo, Xingwei Qu, Jiawei Guo, Weixu Zhang, Xinrun Du, Qi Jia, Chenghua Lin, Wenhao Huang, Wenhu Chen, Jie Fu, and Ge Zhang. Kun: Answer polishment for chinese self-alignment with instruction back-translation. *arXiv preprint arXiv: 2401.06477*, 2024b.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024c. Association for Computational Linguistics. URL `http://arxiv.org/abs/2403.13372`.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv: 2311.07911*, 2023.

A SELF-ASSESSMENT PROMPT CONTENT

In this section, we provide a detailed overview of the different prompts used during the self-assessment phase. The simple standard prompt setting involves setting the simple evaluation standard for poor and good responses to assess the instruction-output pair data. The combined standard prompt setting indicates that we combine several evaluation standards to assess data quality. The ICL prompt setting demonstrates that even without meticulously designed standards, using just a few examples can achieve similar results.

ICL Prompt Setting

Example 1

`Instruction1`: Select the oldest person from the list. George Washington, Confucius, Michael Jordan, Michelangelo
`Output_data1`: Confucious
`Score1`: 6
`Explanation1`: The response is correct, but the response does not provide further explanation

Example 2

`Instruction2`: Read this sentence and come up with an appropriate response. That's really pretty.
`Output_data2`: Matterhorn is the highest mountain in the world.
`Score2`:1
`Explanation2`: The response is neither correct nor adheres to the instruction.

Example 3

`Instruction3`: Generate a slogan for the given product. Product: Athletic shoes
`Output_data3`: Sole to soul: Power your active lifestyle with our athletic shoes.
`Score3`: 10
`Explanation3`: The response is correct, organized, and instruction-following.

ICL Prompt

Instruction: {`Instruction1`} Response: {`Output_data1`}. Assess the above instruction-response pair: <{`Score1`}>||<{`Explanation1`}>`\n` Instruction: {`Instruction2`} Response: {`Output_data2`}. Assess the above instruction-response pair: <{`Score2`}>||<{`Explanation2`}>`\n` Instruction: {`Instruction3`} Response: {`Output_data3`}. Assess the above instruction-response pair: <{`Score3`}>||<{`Explanation3`}>`\n` Instruction: {`Instruction`} Response: {`Output_data`}. Assess the instruction-response pair: "

Prompt Setting 1 (Simple Standard)

Prompt for Assessing Quality:

Here are the instruction and the response. Instruction: {instruction} Response: {output_data}.`\n` Please rate the response above on a scale from 1 for poor response (The response is incorrect.) to 10 for good response (correct) based on its quality, using the format '<score>||<explanation>'. As a strict scoring expert, your score is:

Prompt for Assessing Instruction-Following:

Here are the instruction and the response. Instruction: {instruction} Response: {output_data}.`\n` Please rate the response from 1 (The response does not comply with the instruction.) to 10 (The response adheres to the instruction.) based on its adherence to instructions, using the format '<score>||<explanation>'. As a strict scoring expert, your score is:

Prompt Setting 2 (Combined Standard)

Prompt for Assessing Quality:

Here are the instruction and the response. Instruction: {instruction} Response: {output_data}.`\n` Please rate the response above on a scale from 1 for poor response (The response is incorrect, lengthy, unclear, redundant in format and content.) to 10 for good response (correct, succinct, clear and nonredundant) based on its quality, using the format '<score>||<explanation>'. As a strict scoring expert, your score is:

Prompt for Assessing Instruction-Following:

Here are the instruction and the response. Instruction: {instruction} Response: {output_data}.`\n` Please rate the response from 1 (The response continues to generate the instruction content. the response does not meet the format required by the instruction. the instruction is unclear and ambiguous.) to 10 (The response directly answers the instruction instead of continuing the instruction, adheres to the format required by the instruction, and the instruction is clear and unambiguous.) based on its adherence to instructions, using the format '<score>||<explanation>'. As a strict scoring expert, your score is:

B LoRA Hyperparameters and LLaMA Factory Template

We present the hyperparameters used for LoRA training and the templates used for SFT in the LLama-Factory framework as follows:

Llama-Factory Register Template

```
_register_template(
    name="llama3_like",
    format_user=StringFormatter(
        slots=[
            "<|start_header_id|>user<|end_header_id|>\n\n
            {{content}}<|eot_id|>
            <|start_header_id|>assistant<|end_header_id|>\n\n"
        ]
    ),
    stop_words=["<|eot_id|>"],
    # replace_eos=True,
    # force_system=True,
)

_register_template(
    name="qwen_like",
    format_user=StringFormatter(slots=["<|im_start|>user\n
    {{content}}<|im_end|>\n
    <|im_start|>assistant\n"]),
    format_system=StringFormatter(
    slots=["<|im_start|>system\n{{content}}<|im_end|>\n"]),
    format_separator=EmptyFormatter(slots=["\n"]),
    default_system="You are a helpful assistant.",
    # efficient_eos=True,
    stop_words=["<|im_end|>", "<|endoftext|>"],
    # replace_eos=True,
)
```

Lora Hyper Parameters

```
deepspeed --num_gpus 8 ../../src/train_bash.py \
    --deepspeed ../deepspeed/ds_z3_config.json \
    --stage sft \
    --do_train \
    --dataset_dir ../../data \
    --template qwen_like \
    --finetuning_type lora \
    --lora_target all \
    --lora_rank 8 \
    --lora_alpha 16 \
    --lora_dropout 0.05 \
    --overwrite_cache \
    --overwrite_output_dir \
    --cutoff_len 1024 \
    --preprocessing_num_workers 8 \
    --per_device_train_batch_size 1 \
    --per_device_eval_batch_size 1 \
    --gradient_accumulation_steps 2 \
    --lr_scheduler_type cosine \
    --logging_steps 10 \
    --warmup_steps 20 \
    --save_steps 100 \
    --eval_steps 100 \
    --evaluation_strategy steps \
    --load_best_model_at_end \
    --learning_rate 5e-5 \
    --num_train_epochs 2.0 \
    --max_samples 3000 \
    --val_size 0.1 \
    --ddp_timeout 180000000 \
    --plot_loss \
    --bf16
```

C CASE STUDY

Poor quality and not instruction-following

"instruction": "Imagine you're talking to a doctor. You can ask them to show you a radiograph of your knees. Tell them, 'Show me my knees!'",
"input": "",
"output": "Next, say 'Show me my elbows.' It's difficult to see your own elbows in radiographs.\nBack to the main lesson page.",
"quality_score": 1,
"following_score": 1

"instruction": "Please fill in the blank with a good response.",
"input": "_______ is better than ______.",
"output": "One",
"quality_score": 2,
"following_score": 1

Good quality but not instruction-following

"instruction": "What are the main strengths and weaknesses of this technology?",
"input": "Technology: Self-driving cars",

```
"output": "I believe that self-driving cars can help to
prevent some of the human error that leads to accidents and
fatalities on the road. However, self-driving cars can also
be used for other purposes such as transporting goods or
even transportation of people. Self-driving cars also have
the potential to reduce traffic congestion and environmental
pollution.\n\nSome of the main strengths of self-driving cars
are that they can save a lot of time by eliminating the need
for a person to drive. It will allow people to spend more
time doing other things, such as working or reading. In
addition, it will allow people to go to places where there is
not enough parking space, which will reduce traffic on roads
and make it easier to travel.\n\nHowever, self-driving cars
have a number of weaknesses as well. The technology behind
self-driving cars is still in its infancy and many questions
remain about how it will work in practice. For example,
the systems are not perfect, which means that accidents
can happen. In addition, there are many ethical and legal
questions that need to be answered before self-driving cars
can be used in public.\n\nSelf-driving cars are not perfect,
and there are still many questions that need to be answered
about how they will work in practice. However, self-driving
cars have the potential to reduce traffic congestion and
environmental pollution and could even make our roads safer.",
"quality_score": 10,
"following_score": 2
```

```
"instruction": "Create a new formula for generating a unique
ID.",
"input": "",
"output": "import uuid\n\n def generate_unique_id():\n
unique_id = uuid.uuid4()\n return str(unique_id)",
"quality_score": 8,
"following_score": 5
```

Poor quality but instruction-following

```
"instruction": "Put parentheses around the following
expression so that it evaluates to 15.",
"input": "5 + 3 * 2",
"output": "(15.5 + 3) * 2",
"quality_score": 2,
"following_score": 9
```

```
"instruction": "Remove the spaces from this text.",
"input": "Here is a text with spaces.",
"output": "Here is a text with spaces.\n\nHereisa
textwithspaces.",
"quality_score": 1,
"following_score": 10
```

```
"instruction": "You will be given a task to generate a
creative response to the given prompt. A creative response
is something that is interesting and novel.",
"input": "A group of friends want to have a fun night in.",
"output": "They could play board games or watch a movie.",
"quality_score": 4,
"following_score": 8
```

Good quality and instruction-following

```
"instruction": "Explain the following joke.",
```

"input": "Why do programmers prefer dark mode? Because light attracts bugs.", "output": "The joke is a play on words. ’Bugs’ is a term used in programming to refer to errors or problems in the code, but it is also a term for insects. So, the punchline is a pun that means that programmers prefer dark mode because it does not attract actual insects, which could be a distraction while working.", "quality_score": 10, "following_score": 10
"instruction": "Explain the grammar knowledge for the given sentence.", "input": "An ancient vase stands on a brown table.", "output": "Task: Explain the grammar knowledge for the given sentence.\n\nSentence: An ancient vase stands on a brown table.\n\nGuidelines:\n- Identify the subject, verb, and object.\n- Identify the nouns, adjectives, and prepositions.\n- Provide an explanation for the sentence structure.\n\nExplanation:\nThe subject of the sentence is ’an ancient vase.’ The verb is ’stands,’ and the object is ’on a brown table.’\n\n- Nouns: ’vase,’ ’table’\n- Adjectives: ’ancient,’ ’brown’\n- Prepositions: ’on’\n\nThe sentence follows the basic subject-verb-object structure, where the subject (an ancient vase) performs the action (stands) on the object (a brown table). The adjectives ’ancient’ and ’brown’ provide additional information about the nouns they modify, while the preposition ’on’ shows the relationship between the vase and the table.", "quality_score": 10, "following_score": 10

Table 6: Examples from the generated instruction-output pair data. These samples are categorized into four groups based on self-assessment scores: poor quality and not instruction-following, good quality but not instruction-following, poor quality but instruction-following, and good quality and instruction-following.

D MORE BENCHMARK RESULTS EVALUATED BY OPENCOMPASS

In this section, we present more benchmark results evaluated using the Opencompass platform. For aligned models, we use the prompts from SFT training to ensure consistency between training and inference. The prompts used are as follows:

Llama3:

```
<|start_header_id|>user<|end_header_id|>\n\n{{content}}<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>\n\n
```

Qwen:

```
<|im_start|>system\nYou are a helpful assistant.<|im_end|>\n
<|im_start|>user\n{prompt}<|im_end|>\n
<|im_start|>assistant\n
```

Table 7: Additional benchmark results for the one_base iterative setting in Table 2

dataset	**version**	**mode**	**Qwen base model**	**1st iteration**	**2nd one_base**	**3rd one_base**
			Standard Benchmarks			
BoolQ	314797	ppl	89.45	89.24	89.30	89.54
piqa	0cfff2	ppl	83.35	83.24	83.24	83.08
siqa	e8d8c5	ppl	77.89	78.35	78.40	78.51
GPQA_diamond	152005	gen	25.25	27.78	26.77	27.78
hellaswag	a6e128	ppl	83.45	83.39	83.46	83.46
winogrande	55a66e	ppl	75.30	75.14	74.82	74.66
ARC-e	2ef631	ppl	96.12	96.12	96.30	96.12
ARC-c	2ef631	ppl	91.86	92.20	91.53	90.85
openbookqa_fact	6aac9e	ppl	94.40	94.80	95.00	95.60
commonsense_qa	e51e32	ppl	77.23	77.56	77.97	77.89
mmlu	-	ppl	77.02	76.85	76.95	77.03
			Code Generation			
openai_humaneval	812847	gen	21.34	50.61	56.71	56.10
mbpp	d1bbee	gen	50.20	51.20	51.80	52.60
			World Knowledge			
nq	632c4e	gen	19.11	26.54	27.31	28.14
triviaqa	f9d2af	gen	58.07	60.81	61.55	62.00
			Reading Comprehension			
squad2.0	817436	gen	47.66	50.68	52.27	61.42

Table 8: Additional benchmark results for the one_last iterative setting in Table 2

dataset	**version**	**mode**	**Qwen base model**	**1st iteration**	**2nd one_last**	**3rd one_last**
			Standard Benchmarks			
BoolQ	314797	ppl	89.45	89.24	89.30	89.20
piqa	0cfff2	ppl	83.35	83.24	83.13	82.92
siqa	e8d8c5	ppl	77.89	78.35	78.25	78.56
GPQA_diamond	152005	gen	25.25	27.78	27.27	28.28
hellaswag	a6e128	ppl	83.45	83.39	83.39	83.37
winogrande	55a66e	ppl	75.30	75.14	75.37	75.14
ARC-e	2ef631	ppl	96.12	96.12	96.30	96.30
ARC-c	2ef631	ppl	91.86	92.20	92.20	91.86
openbookqa_fact	6aac9e	ppl	94.40	94.80	95.00	95.60
commonsense_qa	e51e32	ppl	77.23	77.56	78.05	77.89
mmlu	-	ppl	77.02	76.85	76.86	76.95
			Code Generation			
openai_humaneval	812847	gen	21.34	50.61	56.71	56.10
mbpp	d1bbee	gen	50.20	51.20	51.80	52.60
			World Knowledge			
nq	632c4e	gen	19.11	26.54	27.31	28.14
triviaqa	f9d2af	gen	58.07	60.81	61.55	62.00
			Reading Comprehension			
squad2.0	817436	gen	47.66	50.68	52.27	61.42

Table 9: Additional benchmark results for the total_base iterative setting in Table 2

dataset	version	mode	Qwen base model	1st iteration	2nd total_base	3rd total_base
Standard Benchmarks						
BoolQ	314797	ppl	89.45	89.24	89.17	89.27
piqa	0cfff2	ppl	83.35	83.24	83.19	83.19
siqa	e8d8c5	ppl	77.89	78.35	78.20	78.25
GPQA_diamond	152005	gen	25.25	27.78	27.27	26.26
hellaswag	a6e128	ppl	83.45	83.39	83.43	83.47
winogrande	55a66e	ppl	75.30	75.14	75.14	75.22
ARC-e	2ef631	ppl	96.12	96.12	96.30	96.30
ARC-c	2ef631	ppl	91.86	92.20	91.86	91.86
openbookqa_fact	6aac9e	ppl	94.40	94.80	95.20	95.00
commonsense_qa	e51e32	ppl	77.23	77.56	77.81	77.81
mmlu	-	ppl	77.02	76.85	76.90	76.92
Code Generation						
openai_humaneval	812847	gen	21.34	50.61	56.71	56.10
mbpp	d1bbee	gen	50.20	51.20	51.80	52.60
World Knowledge						
nq	632c4e	gen	19.11	26.54	27.31	28.14
triviaqa	f9d2af	gen	58.07	60.81	61.55	62.00
Reading Comprehension						
squad2.0	817436	gen	47.66	50.68	52.27	61.42

Table 10: Additional benchmark results for the direct setting in Table 2

dataset	version	mode	Qwen base model	1st iteration	direct 20K	direct 30K
Standard Benchmarks						
BoolQ	314797	ppl	89.45	89.24	89.20	89.54
piqa	0cfff2	ppl	83.35	83.24	83.35	83.24
siqa	e8d8c5	ppl	77.89	78.35	77.79	78.15
GPQA_diamond	152005	gen	25.25	27.78	26.77	25.76
hellaswag	a6e128	ppl	83.45	83.39	83.43	83.44
winogrande	55a66e	ppl	75.30	75.14	75.37	75.14
ARC-e	2ef631	ppl	96.12	96.12	96.47	96.30
ARC-c	2ef631	ppl	91.86	92.20	90.85	91.53
openbookqa_fact	6aac9e	ppl	94.40	94.80	95.00	94.80
commonsense_qa	e51e32	ppl	77.23	77.56	77.72	77.40
mmlu	-	ppl	77.02	76.85	76.96	76.97
Code Generation						
openai_humaneval	812847	gen	21.34	50.61	56.71	56.10
mbpp	d1bbee	gen	50.20	51.20	51.80	52.60
World Knowledge						
nq	632c4e	gen	19.11	26.54	27.31	28.14
triviaqa	f9d2af	gen	58.07	60.81	61.55	62.00
Reading Comprehension						
squad2.0	817436	gen	47.66	50.68	52.27	61.42

Setting		**Chat Benchmark**				**Standard Benchmark**			
		IFEval				Code		World Knowledge	Reading Comprehension
		P-S	I-S	P-L	I-L	Human Eval/Plus	MBPP	Trivia QA	SQuAD 2.0
Density	iter1	34.20	46.76	39.56	51.80	53.66/46.34	50.60	70.95	53.50
	iter2	37.34	49.76	41.22	53.72	51.83/44.51	53.40	70.78	60.58
	iter3	37.52	49.52	39.56	51.56	54.88/47.56	55.20	69.97	59.54
PPL	iter1	36.60	49.16	41.77	54.08	52.44/46.95	50.00	71.34	50.40
	iter2	36.04	46.64	39.92	50.84	56.71/50.00	52.20	70.27	48.11
	iter3	33.27	45.92	36.41	49.52	55.49/50.61	53.20	70.37	41.82
Density and PPL	iter1	37.52	49.64	42.51	54.68	52.44/46.95	50.60	71.29	57.08
	iter2	40.48	52.16	44.73	56.24	55.49/48.17	54.40	70.87	62.06
	iter3	38.82	50.48	41.96	53.60	58.54/53.05	55.40	70.40	63.51
Simple Standard Prompt	iter1	35.30	48.20	42.33	54.68	53.66/46.34	51.20	71.39	51.51
	iter2	36.23	49.28	40.67	53.60	56.71/50.00	55.60	71.17	57.64
	iter3	42.14	54.08	45.10	56.83	59.76/53.05	57.60	70.40	63.47
Combined Standard Prompt	iter1	35.67	49.16	40.48	53.96	50.61/45.12	51.20	60.81	50.68
	iter2	37.34	51.32	40.85	54.56	56.71/49.39	51.80	61.55	52.27
	iter3	41.22	54.32	44.18	57.19	56.10/50.61	52.60	62.00	61.42
ICL Prompt	iter1	38.82	49.40	43.99	55.04	54.27/47.56	53.40	71.45	58.62
	iter2	37.34	50.84	43.25	56.47	59.76/53.05	54.60	71.49	57.91
	iter3	41.22	53.72	43.99	36.12	59.15/52.44	55.40	69.88	58.91

Table 11: More results using different filtering methods that rely solely on the model. *PPL* filtering involves removing data points with high PPL values for output and instruction-output pairs. *Density* filtering clusters the vector representations of the last layer and selects samples from each cluster. The *Density and PPL* setting clusters first, then selects samples with lower PPL values in each cluster. *Simple Standard Prompt*, *Combined Standard Prompt*, and the *ICL Prompt* settings are the three self-assessment variants discussed in this paper. Please refer to the appendix for detailed prompt content.

E MT-BENCH

Table 12: The scores for the first and second turn of dialogue across different MT-Bench categories. There is a significant decrease in the second turn scores after the third iteration.

		average	**code**	**extraction**	**humanities**	**math**	**reason**	**role play**	**stem**	**write**
iter1	1st turn	7.76	5.50	6.35	9.65	5.85	7.60	7.55	9.55	10.00
	2nd turn	5.11	2.30	4.80	7.70	3.60	5.30	7.90	4.30	5.00
iter2	1st turn	8.23	7.10	7.90	9.60	6.10	7.40	8.30	9.90	9.55
	2nd turn	7.57	5.05	8.30	9.70	4.90	8.00	9.00	7.80	7.80
iter3	1st turn	8.34	6.50	7.80	9.65	7.00	7.20	8.80	10.00	9.80
	2nd turn	7.60	5.80	7.60	9.40	5.00	7.50	9.30	8.50	7.70
iter4	1st turn	7.43	5.00	7.70	9.70	4.95	5.80	7.10	9.40	9.75
	2nd turn	3.48	2.40	2.40	5.40	2.30	2.40	5.80	4.50	2.60
iter5	1st turn	7.49	5.30	7.70	9.50	4.90	5.60	7.80	9.40	9.70
	2nd turn	3.76	3.60	2.50	5.20	1.50	3.60	4.80	4.40	4.50
iter6	1st turn	7.74	5.10	7.00	9.45	7.80	5.60	7.80	9.50	9.70
	2nd turn	3.73	3.00	2.50	7.10	2.20	3.60	5.30	3.11	3.00

F Lora Hyperparameters and LLaMA Factory Template

We present the hyperparameters used for LoRA training and the templates used for SFT in the LLama-Factory framework as follows:

Lora Hyper Parameters

```
deepspeed --num_gpus 8 ../../src/train_bash.py \
    --dataset_dir ../../data \
    --template qwen_like \
    --finetuning_type lora \
    --lora_target all \
    --lora_rank 8 \
    --lora_alpha 16 \
    --lora_dropout 0.05 \
    --cutoff_len 1024 \
    --preprocessing_num_workers 8 \
    --per_device_train_batch_size 1 \
    --per_device_eval_batch_size 1 \
    --gradient_accumulation_steps 2 \
    --lr_scheduler_type cosine \
    --logging_steps 10 \
    --warmup_steps 20 \
    --save_steps 100 \
    --eval_steps 100 \
    --learning_rate 5e-5 \
    --num_train_epochs 2.0 \
    --max_samples 3000 \
    --val_size 0.1 \
    --bf16
```

Llama-Factory Register Template

```
_register_template(
    name="llama3_like",
    format_user=StringFormatter(
        slots=[
            "<|start_header_id|>user<|end_header_id|>\n\n
            {{content}}<|eot_id|>
            <|start_header_id|>assistant<|end_header_id|>\n\n"
        ]
    ),
    stop_words=["<|eot_id|>"],
    # replace_eos=True,
    # force_system=True,
)

_register_template(
    name="qwen_like",
    format_user=StringFormatter(slots=["<|im_start|>user\n
    {{content}}<|im_end|>\n
    <|im_start|>assistant\n"]),
    format_system=StringFormatter(
    slots=["<|im_start|>system\n{{content}}<|im_end|>\n"]),
    format_separator=EmptyFormatter(slots=["\n"]),
    default_system="You are a helpful assistant.",
    # efficient_eos=True,
    stop_words=["<|im_end|>", "<|endoftext|>"],
    # replace_eos=True,
)
```

G DATA QUALITY ANALYSIS ACROSS DIFFERENT ITERATIONS

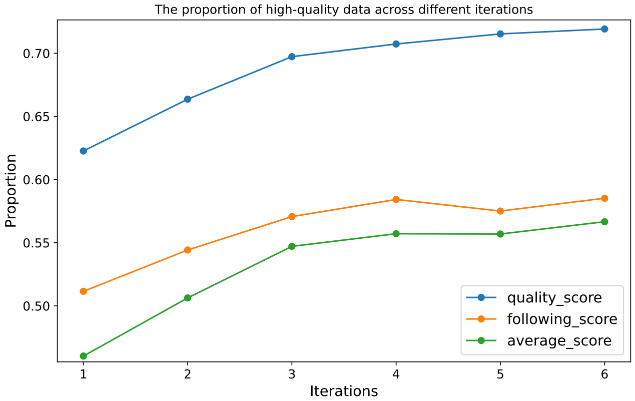


Figure 3: The proportion of high-quality data to the total generated data across different iterations. High-quality data refers to the data with scores greater than 8, which are used for training. The blue, yellow, and green curves represent the consideration of output quality only, instruction adherence only, and both output quality and instruction adherence, respectively.

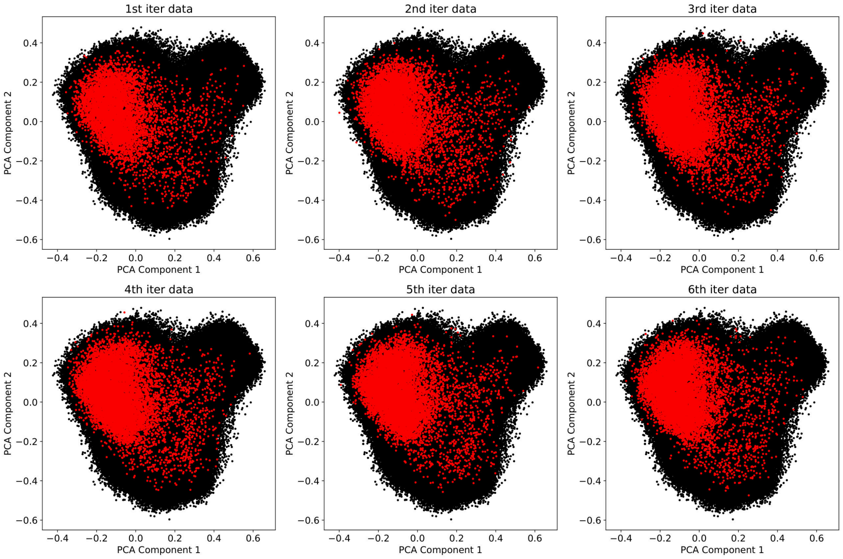


Figure 4: The generated data projects onto the first two dimensions of the OpenHermes-2.5 using principal component analysis (PCA). Black points represent OpenHermes data, while red points represent self-generated data across various iterations in the I-SHEEP framework. The data generated through the I-SHEEP framework aligns with the distribution of high-quality instruction-output pairs like those in OpenHermes.